Amendments To The Specification:

On page 10, please delete the paragraph beginning at line 30, and replace it with the following paragraph:

Register 22 is also preferably a 32-bit register. As is described below, bits 0-15 of register 22 are used to set a release bit L(x) for each of the shared resources ~~registers~~, which releases the lock on the particular resource by the processor that currently holds the lock. Similar to register 20, for a particular shared resource Rx, where x represents a particular shared resource, bit 0 of register 22 holds the release bit for shared resource R0; bit 1 of register 22 holds the release bit for shared resource R1 and so on. Also similar to register 20, register 22 is capable of supporting up to 16 shared resources, particularly with bits 0 through 15 of the register. It will be understood that the size of register 22 may be adjusted to accommodate more shared resources.

On page 11, please delete the paragraph beginning at line 12, and replace it with the following paragraph:

Associated with each of the bits 0 through 15 of register 20 are lock request arbiters 24a-24m. Each of these lock request arbiters 24a-24m receive lock bits from each of the processors P0, P1, P2...P(m-1) that request access to a particular resource R0, R1...R(n-1). Fig. 2 is a schematic block diagram of one of the arbiters 24. Arbiter 24 includes an AND gate 26 for receiving lock bits from the processors P0, P1, P2...P(m-1) via the CPU bus 16 on line 27. An encoder or demultiplexer 28 receives the output of the AND gate 26 and outputs the input on one of lines 30a-30d, which correspond to each of the layers 23a-23d of upper register 21a. A selection signal on write select line 32 determines which of the layers the incoming lock bit will be written to. Shown at 33a-33d are each of the lock bits Q_{P0}, Q_{P1}, Q_{P2}, $Q_{P(m-1)}$, respectively, that correspond to the bit of upper register 21a associated with a particular shared resource R0, R1...R(n-1). The default setting for bits Q_{P0}, Q_{P1}, Q_{P2}, $Q_{P(m-1)}$ is "0", unless a "1" is written to the bit by the demux 28, as described below. Multiplexer 34 receives bits Q_{P0}, Q_{P1}, Q_{P2}, $Q_{P(m-1)}$ as inputs and outputs one of the bits to the CPU bus 16 via line 35. A selection signal on

read select line 36 determines which of the lock bits will be read out from multiplexer 34. An OR gate 37 receives each of the bits Q_{P0}, Q_{P1}, Q_{P2}, $Q_{P(m-1)}$ and outputs the logic result, an access arbitration signal, on line 38. An inverter 39 receives the output of the OR gate 37 and provides the inverted signal to the AND gate 26. Alternatively, OR gate 37 and inverter 39 could be replaced with a NOR gate.

On page 14, please delete the paragraph beginning at line 7, and replace it with the following paragraph:

In the event that a processor that has a lock on a shared resource exceeds a predetermined communication time or timeout, any of the other (non-locked) processors can set a release bit L of ~~the bit of~~ register 22 that corresponds to the particular shared resource that has been locked for longer than the timeout period. Setting the release bit L causes all of the lock bits Q_{P0}, Q_{P1}, Q_{P2}, $Q_{P(m-1)}$ stored in bits 33a-33d to be cleared to "0", thus releasing the processor that was lock beyond the timeout period.

On page 15, please delete the paragraph beginning at line 27, and replace it with the following paragraph:

If, when processor P0 attempts to set its lock bit in Step 42, another processor has already set its lock bit, Step 44, the arbiter 24 blocks the lock bit from processor P0, as described above, Step 46. The status bit S (bit 17) is set to one because the resource R1 is locked, Step 48. Accordingly, when processor P0 reads the status bit S and its own lock bit Q_{P0} for the resource, Step 50, it reads that the status bit S is set to 1, Step 52, but that its own request bit Q_{P0} is set to zero, Step 54. At this point, the process enters monitoring loops to monitor the status of the resource R1. In Step 62, the processor P0 monitors a timeout which begins when the processor accessing the resource R1 begins its access. It will be understood that the length of the timeout is set to accommodate the needs of the specific system in which the controller operates. If the timeout has [[no]] not expired, Step 64, the processor continues to monitor the timeout. If the timeout has expired, Step 64, the processor P0 sets the release bit L for the resource R1, Step 70, which forces the locked processor to be released from its access to the resource R1, as

described above, Step 72. The process returns to Step 42 where the processor P0 is able to compete for access to the resource R1.